IBM Research

Parallel Filesystem

C. Howson

Feb. 24, 2005

Outline

- Parallel Filesystems on BlueGene/L
- Storage subsystems
- NFS
- GPFS
- ...
- Performance

Characteristics of Parallel Filesystems

- MPI-IO is most common client of parallel filesystem
  - Collective IO is very common, all nodes write to a different portion of same file
- Streaming IO all the way to disks is important
  - IO node ram is small, not much opportunity to cache
  - Aggregate ram is large, fileservers' cache may be too small as well
- Everything goes through GigE network in BlueGene/L
  - IO node is relatively underpowered, FS overhead will lower bandwidth

Storage Subsystems

- BlueGene/L is big, fileserver had better be able to support huge systems
  - Similar philosophy of many low cost disks
- Disk streaming performance is important
  - 15krpm U320 SCSI = ~20MB/s
  - 10krpm U320 SCSI = ~15MB/s
  - 7200rpm SATA = ???
- Metrics
  - Disks/U of rack space
  - Fileserver network bandwidth
    - IO nodes/Fileserver
    - Fileserver/Disks
- In many cases, fileserver is already there, BlueGene/L must support

Parallel Filesystems on BlueGene/L

- NFS
  - Simple, ubiquitous, relatively fast
  - Hybrid possible: BG/L nfs mounts from parallel fileserver
  - Poor support for shared file data between clients (MPI-IO)
- GPFS
  - Fully parallel filesystem: client writes directly to fileserver node with disk
  - Prototype runs on BlueGene/L, but needs tuning
- Others
  - Pvfs2
  - Lustre

GPFS Architecture

High capacity:

- Large number of disks in a single FS

High BW access to single file

- Large block size, full-stride I/O to RAID
- Wide striping – one file over all disks
- Multiple nodes read/write in parallel

High availability

- Nodes: log recovery restores consistency after a node failure
- Data: RAID or internal replication
- On-line management (add/remove disks or nodes without un-mounting)

Single-system image, standard POSIX interface

- Distributed locking for read/write semantics

GPFS Distributed Locking

- Distributed locking essential to …
  - synchronize file system operations for POSIX semantics,
  - synchronize updates to file system metadata on disk to prevent corruption,
  - maintain cache consistency of data and metadata cached on different nodes.
- Synchronization requires communication …
  - Problem: sending a lock message for every operation will not scale.
  - Solution: Token-based lock manager allows “lock caching”.

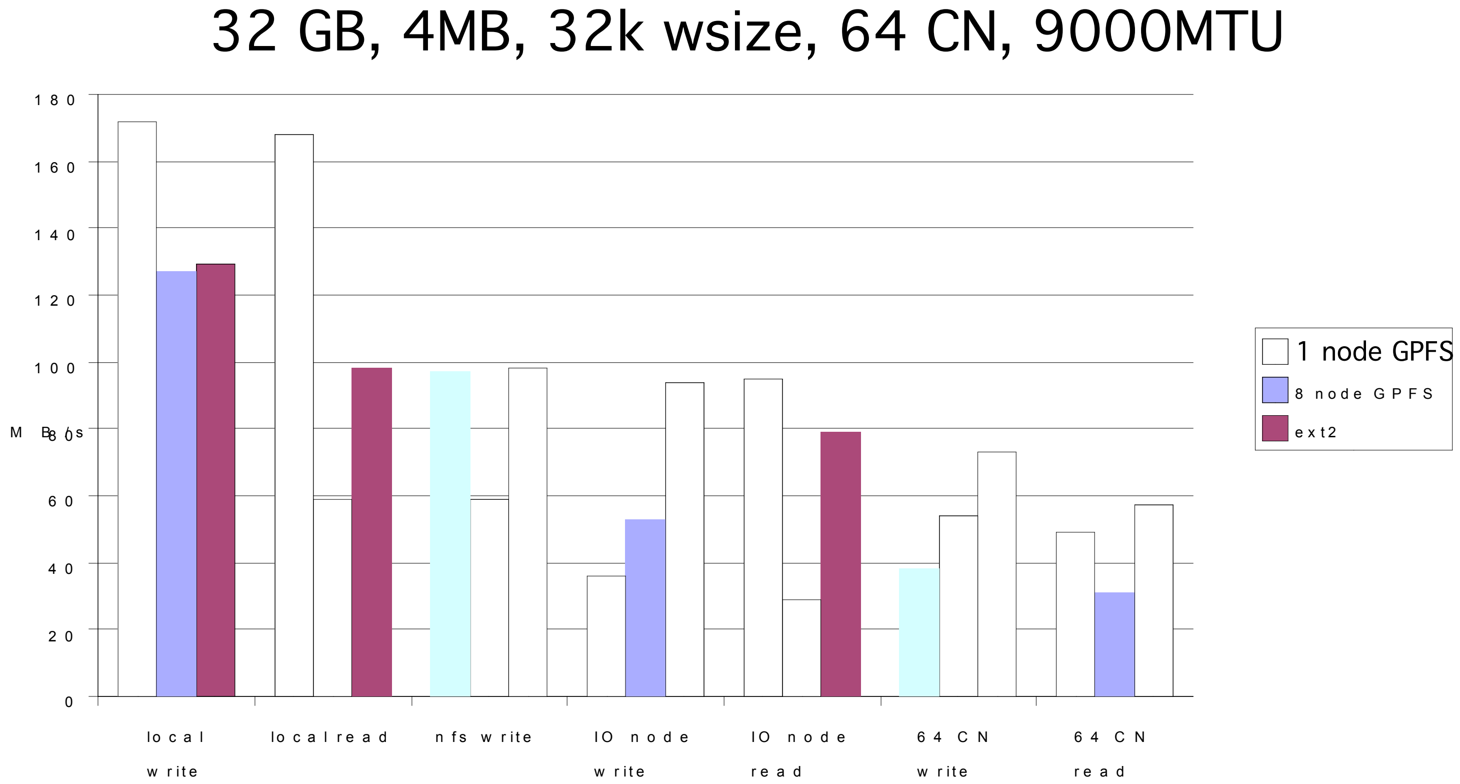
GPFS Token Based Locking

- Token server grants tokens.
- Token represents right to read, cache, and/or update a particular piece of data or metadata.
- Single message to token server allows repeated access to the same object.
- Conflicting operation on another node will revoke the token.
- Force-on-steal: dirty data & metadata flushed to disk when token is stolen.

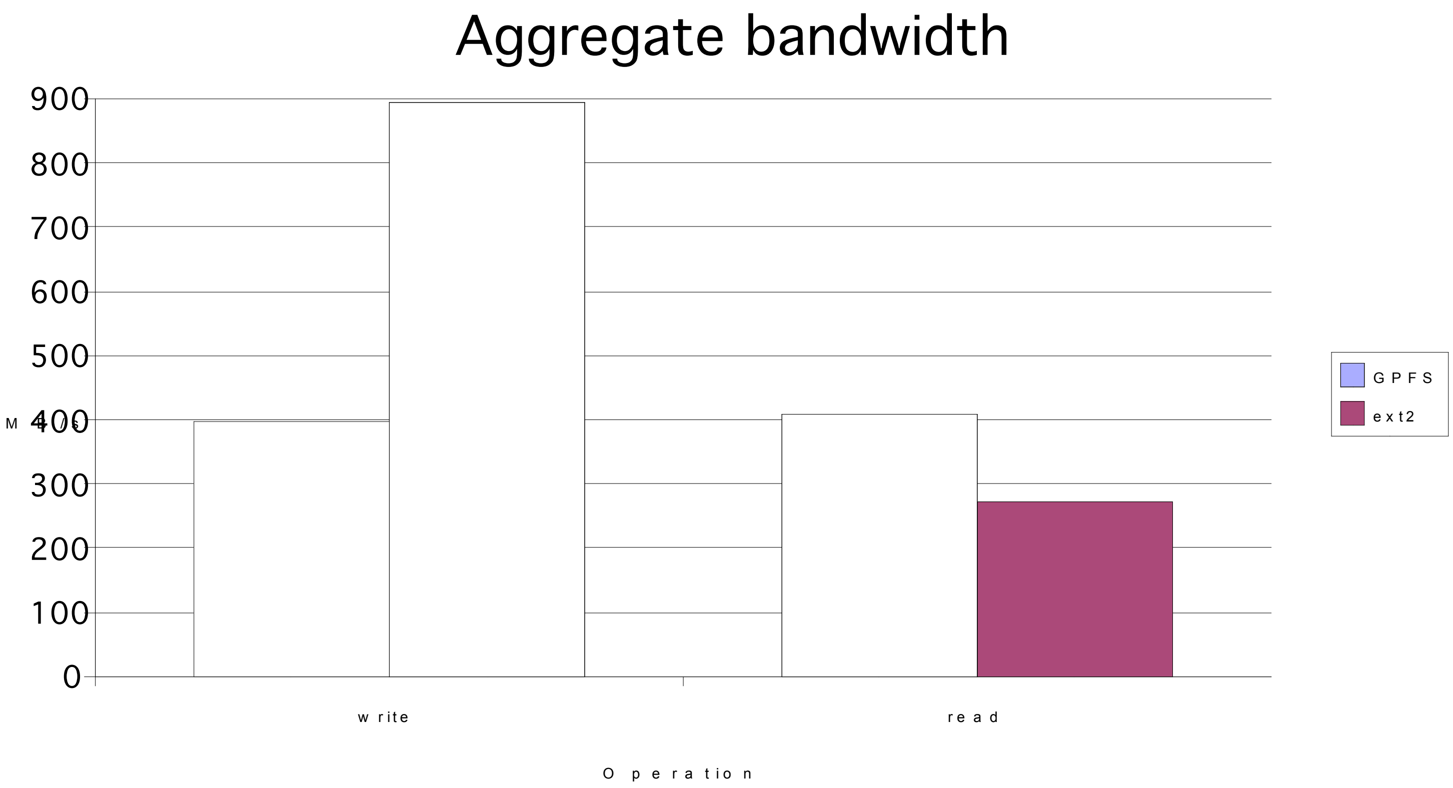
GPFS on BlueGene/L

- GPFS has client and manager nodes
  - Client only deals with own IO requirements, manager does token management as well
  - IO node is gpfs client
- GPFS consists of kernel module and user level daemon
  - kernel module handles local fs related functions
  - user level daemon handles external communications
- Initial prototype works
  - Performance is slow, due to debug build?
  - Need slight kernel modifications to support user level daemon

NFS client to GPFS or ext2 server



64 IO nodes NFS clients to 8 node GPFS or ext2 server



Conclusion

- BlueGene/L is high performance, so it needs a high performance filesystem
- Fileserver needs to scale up to large number of clients, servers, disks
- IO nodes don't have much RAM or computational power
- Tuning system parameters is very important, application dependent
  - NFS - rsize, wsize, tcp, udp, async, {r,w}mem_default,...
  - GPFS - pagepool, blocksize